REMARKS

Claims 1-37, and 39-41 are pending in the present application. By this Response, claims 1, 13 and 25 are amended to recite the features of identifying one or more methods invoked by a property editor and selecting a graphical user interface based on the one or more methods invoked by the property editor. These features were originally presented in original claims 2, 14 and 26. In addition, claims 2, 14 and 26 are amended to recite that the one or more methods invoked by the property editor identify one or more abilities of the property editor. Further, claims 3, 4, 7, 9, 15, 16, 19, 21, 27, 28, 31 and 33 are amended to correct their dependency in view of the amendments to claims 1, 2, 13, 14, 25 and 26. Claim 37 is amended to incorporate the subject matter of claim 38 and to recite that the property is a data type. Claim 38 is canceled accordingly. Claims 39-41 are amended to correct their dependency to depend from claim 37. Reconsideration of the claims in view of the above amendments and the following remarks is respectfully requested.

I. 35 U.S.C. § 102, Alleged Anticipation of Claims 1-3, 7-9, 12-15, 19-21, 24-27, 31-33, 36-38 and 40-41

The Office Action rejects claims 1-3, 7-9, 12-15, 19-21, 24-27, 31-33, 36-38 and 40-41 under 35 U.S.C. § 102(e) as being allegedly anticipated by Zimmerman et al. (U.S. Patent No. 6,417,872). This rejection is respectfully traversed.

With regard to claim 1, the Office Action states:

> As per claim 1, Zimmerman et al. ("Zimmerman") teaches a method of editing a property, comprising:
>
> identifying one or more abilities of a property editor (see Zimmerman, column 7, line 66 – column 8, line 2; the examiner interprets the type of the property to be edited as an ability of the property editor because it determines the actions the property editor is able to perform);
>
> selecting a graphical user interface based on the one or more abilities of the property editor (see Zimmerman, column 9, lines 7-11); and

providing the graphical user interface for use in editing the property (see Zimmerman, column 9, lines 11-17).

Office Action dated January 29, 2004, page 3.

Claim 1, which is representative of claims 13 and 25 with regard to similarly recited subject matter, reads as follows:

> 1. A method, in a data processing system, for editing a property, comprising:
> identifying one or more methods invoked by a property editor associated with the property;
> selecting a graphical user interface based on the one or more methods invoked by the property editor; and
> providing the graphical user interface for use in editing the property. (emphasis added)

Zimmerman is directed to a system for adding application defined properties and application defined property sheet pages to a list of system defined properties. Once added, the application defined properties may be displayed and edited. In addition, a user may select several objects, display the properties common to all of the objects in a list, and edit the common properties. Further, a user may select several objects, display the property sheet pages common to all of the objects, and edit the properties on these property sheet pages. Also, a user may switch between viewing a property in a list of properties and viewing a property on a property sheet page.

Thus, Zimmerman is concerned with three main objectives. The first objective is providing a system that allows a user to view and edit application defined properties as well as system defined properties. The second objective Zimmerman is concerned with is allowing a user to view and edit several objects, having common properties, simultaneously. Finally, the third objective of Zimmerman is to provide a system that allows a user to switch between per-property browsing and property sheet page methods. While Zimmerman may speak generally of editing object properties, Zimmerman does not teach identifying one or more methods associated with a property editor. Similarly, Zimmerman does not teach selecting a graphical user interface based on the one or more methods associated with the property editor.

In other words, claims 1, 13 and 25 recite selecting a graphical user interface, to edit properties, based on the methods invoked by a particular property editor for the

property. Zimmerman does not teach such a feature. To the contrary, in Zimmerman, the graphical user interface is selected by the user when navigating between a per-property browser and a property sheet page (column 7, lines 49-63). This feature is described in more detail at column 8, line 35 - column 9, lines 22 of Zimmerman, which reads as follows:

> Fig. 9 is a flowchart of the steps performed by the per-property browser when either the drop down list button or the property sheet page map button have been activated. First, the per-property browser receives user input which indicates which step the per-property browser should take next (step **80**). Then the per-property browser determines whether the user input is a request to activate a drop down list button, which indicates a desire to view the drop down list (step **82**). If the user input requests activation of the drop down list button, then the per-property browser calls the GetPredefinedStrings ( ) function to obtain character strings corresponding to the possible values of the property (step **84**). Then the drop down list is displayed (step **86**). If the user input is not a request to activate the drop down list button, then the per-property browser determines if the user input is a request to select an element in a drop down list which is already displayed (step **88**). If the user input is a request to select an element, then the per-property browser calls the GetPredefinedValue ( ) function to obtain the value corresponding to the user's choice (step **90**). Then this value replaces the current property value (step **91**). For example, if the per-property browser receives user input requesting activation of the drop down list button for a color property, which allows selection of a color for text, then the per-property browser calls the GetPredefinedStrings ( ) function. The GetPredefinedStrings ( ) function may return character strings such as "Red," "Blue," and "Green," which corresponds to possible values of the color property. When the user selects one of these character strings, such as "Red," the per-property browser calls the GetPredefinedValue ( ) function. The GetPredefinedValue ( ) function returns the actual value corresponding to the selected character string, such as an RGB value for the selected color "red."
>
> Continuing with the flowchart, if the user input is not a request to select an element, then the per-property browser determines if the user input is a request to activate the property sheet page map button (step **92**). If the user input is a request to activate a property sheet page map button, then the user desires to view the currently selected property on a property sheet page. The per-property browser calls the MapPropertyToPage ( ) function to display the currently selected property on a property sheet page (step **93**). In particular, the per-property browser calls the MapPropertyToPage ( ) function to obtain an application defined property

sheet page identifier for a property sheet page. Then, the per-property browser invokes the property sheet page browser, passing to it the class identifier returned by the MapPropertyToPage ( ) function.

Thus, in Zimmerman, the user can view a graphical user interface containing a drop down list. In addition, the user can view a property sheet page graphical user interface by activating a property sheet page map button. Although the property sheet page graphical user interface may be selected by the user by activating a property sheet page map button, the graphical user interface is selected to switch between a property sheet page graphical user interface and a graphical user interface containing a drop down list of properties. In other words, the graphical user interface is not selected based on the methods invoked by a property editor associated with a property that is to be edited but rather, on the selections made by the user.

The present invention permits a property and an associated property editor to be defined. Different graphical user interfaces may be provided based on the different methods that may be invoked by property editors. The present invention determines what methods are invoked by the property editor for the property that is to be edited and then, based on the identification of these methods, a graphical user interface may be selected for graphically representing the property editor. Zimmerman does not provide such functionality. To the contrary, in Zimmerman, regardless of the type of property or property editor that may be associated with the property, a user may select either the property sheet page graphical user interface or the drop down list of properties graphical user interface. The graphical user interface that is selected has no relation to the property editors for the properties or the methods invoked by the property editors of the properties.

With regard to independent claim 37, the Office Action states that the rejection is based on the same rationale as claim 2 (which is dependent from claim 1), because claim 2 and claim 37 are of similar scope. Thus, similar to claim 1, claim 37 recites that a graphical user interface is selected based on one or more methods invoked by a property editor for the property. As discussed above, Zimmerman does not teach this feature.

In view of the above, Applicants respectfully submit that Zimmerman does not teach each and every feature of independent claims 1, 13, 25 and 37 as required under 35 U.S.C. § 102(e). At least by virtue of their dependency on claims 1, 13, 25 and 37, Applicants respectfully submit that Zimmerman does not teach each and every feature of

dependent claims 2-3, 7-9, 12, 14-15, 19-21, 24, 26-27, 31-33, and 36. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1-3, 7-9, 12-15, 19-21, 24-27, 31-33 and 36-37 under 35 U.S.C. § 102(e).

Furthermore, Zimmerman does not teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. Absent the Examiner pointing out some teaching or incentive to implement Zimmerman in selecting a graphical user interface based on one or more methods invoked by a property editor of a property to be edited, one of ordinary skill in the art would not be led to modify Zimmerman to reach the present invention when the reference is examined as a whole. Absent some teaching, suggestion, or incentive to modify Zimmerman in this manner, the presently claimed invention can be reached only through an improper use of hindsight using the Applicants' disclosure as a template to make the necessary changes to reach the claimed invention.

Moreover, in addition to being dependent on claims 1, 13 and 25, claims 3, 7-9, 12, 15, 19-21, 24, 27, 31-33, 36 and 40-41 are patentable over Zimmerman by virtue of the specific features recited therein. For example, with regard to claims 7, 8, 19, 20, 31 and 32, Zimmerman does not teach that if one or more abilities of the property editor include an ability to edit a property using tags, the graphical user interface includes at least one of a popup choice selection area virtual button and a current selection display field. The Office Action alleges that this feature is taught at column 6, lines 26-32 which reads:

> FIG. 5 is an example of a user interface in which a per-property browsing drop down list 37 is displayed as it appears in the preferred embodiment of the invention. A drop down list 37 is displayed when the drop down list button 30 has been activated for a particular property. The drop down list 37 shows other possible values for the property.

While this section of Zimmerman teaches the use of a drop down list button 30 and the displaying of a drop down list when the button 30 is activated, there is nothing in this, or any other, section of Zimmerman that teaches, or even suggests, that this drop down list button 30 is provided when the one or more methods invoked by a property editor associated with a property to be edited identify an ability of the property editor is as being able to edit a property using tags. Thus, while Zimmerman may teach a drop

down list, Zimmerman still does not teach or suggest the specific features of claims 7, 8, 19, 20, 31 and 32.

With regard to claims 9, 21 and 33, Zimmerman does not teach that if the one or more abilities include an ability to edit the property using a custom editor interface, the graphical user interface includes a popup custom component area virtual button. The Office Action alleges this feature is taught at column 6, lines 53-64 of Zimmerman, which reads as follows:

> A property sheet page browser 42 is within a window 40. The property sheet page browser 42 is part of the development environment, and it provides the framework within which an application defined property sheet page 44 displays itself. The property sheet page browser 42 contains a tab 45 which identifies a property group. For instance, tab 45 indicates that the property sheet page is the "Font" property group. Other tabs 46 are also displayed alongside tab 45 in such a way that the display resembles a set of tabbed index cards. The other tabs 46 represent other property groups associated with a particular object. The application defined property sheet page 44 contains various properties of an object which may be displayed and edited.

This section teaches selecting a tab in a property sheet browser to view property groups associated with an object. The Examiner interprets the tabs representing the property groups in Zimmerman to be a popup custom component area virtual button because by selecting a tab, a custom property sheet page is allegedly displayed (Office Action, page 4). While a property sheet page containing application defined properties may be displayed, this section does not teach a custom editor. The present specification provides an exemplary definition of a custom editor as an editor that uses methods created by the programmer rather than defined by the programming language (Page 14, lines 9-16). A custom property editor is simply not taught in Zimmerman. Nowhere is anything mentioned in Zimmerman regarding a programmer defining methods to edit properties. Thus, as Zimmerman does not teach a custom editor, Zimmerman cannot teach a custom editor interface.

Furthermore, there is no teaching in Zimmerman as to the identification of one or more methods invoked by a property editor that identifying abilities of the property editor as including the ability of editing a property using a customer editor interface and, based

on this identification, providing a graphical user interface that includes a popup custom component area virtual button, as recited in claims 9, 21 and 33. Thus, despite the allegations made by the Office Action, the Zimmerman reference, in actuality, does not teach or even suggest the specific features recited in claims 9, 21 and 33. This same distinction applies to claims 12, 24 and 36 which depend from claims 9, 21 and 33, respectively.

Regarding claim 41, Zimmerman does not teach that the one or more methods include at least one of a supportsCustomEditor method and a getCustomEditor method. The Office Action alleges the "MapPropertyToPage()" function in Zimmerman is synonymous to the getCustomEditor method in the present invention. Applicants respectfully disagree. For the same reasons as noted above, Zimmerman does not teach using a custom editor. In addition, the "MapPropertyToPage()" function enables switching from a per-property browsing list to a property sheet page containing the particular property (Zimmerman, column 8, lines 20-23). This has nothing to do with supporting a custom editor as does the supportsCustomEditor method and the getCustomEditor method of the present invention. As described in Table 1 of the present specification, the getCustomEditor() is a method that permits a full customer component that edits the property to be made available and the supportsCustomEditor method determines whether the property editor supports a custom editor. Neither of these functions is performed by the "MapPropertyToPage()" function of Zimmerman.

II. 35 U.S.C. § 103, Alleged Obviousness of Claims 4-6, 10, 11, 16-18, 22, 23, 28-30, 34, 35 and 39

The Office Action rejects claims 4-6, 10, 11, 16-18, 22, 23, 28-30, 34, 35 and 39 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Zimmerman (U.S. Patent No. 6,417,872) in view of Lindhorst et al. (U.S. Patent No. 6,337,696). This rejection is respectfully traversed for at least the same reasons as noted above with regard to claims 1, 13 and 25 from which claims 4-6, 10, 11, 16-18, 22, 23, 28-30, 34, 35 and 39 depend, respectively.

Specifically, Zimmerman does not teach identifying one or more methods invoked by a property editor for the property that is to be edited. Similarly, Zimmerman does not teach selecting a graphical user interface based on the one or more methods invoked by the property editor. In addition, Lindhorst does not provide for the deficiencies of Zimmerman. That is, neither Zimmerman nor Lindhorst, either alone or in combination, teach or suggest identifying one or more methods invoked a property editor for a property that is to be edited or selecting a graphical user interface based on the one or more methods invoked by the property editor.

Lindhorst is directed to a method for creating and editing event handlers that link events triggered on one object to take actions on one or more different objects. A graphical user interface contains three different panes, an event pane, an action pane and a code pane. A user selects an event icon in an event pane to link that event to a desired action in the action pane. The generated code can then be viewed in the code pane.

In column 18, lines 18-68, Lindhorst discusses editing of properties using one of a "String Dialog Box," a "Color Dialog Box," and a "Number Dialog Box" based on the type of property determined at state 410 in Figure 6 of Lindhorst. Thus, while Lindhorst discloses a different editor interface for editing the value of a property based on the identified property type, Lindhorst still does not teach or suggest identifying one or more methods invoked by a property editor associated with a property to be edited and then selecting a graphical user interface based on the one or more methods invoked by the property editor. To the contrary, the Lindhorst reference merely identifies a type of the property and determines a dialog box to be displayed based on the type of property. Thus, any alleged combination of Lindhorst and Zimmerman, even if such a combination were somehow possible and one were motivated to attempt it, still would not result in the invention recited in independent claims 1, 13, 25 and 37.

In view of the above, Applicants respectfully submit that neither Zimmerman nor Lindhorst, either alone or in combination, teach or suggest each and every feature of independent claims 1, 13, 25 and 37. At least by virtue of their dependency on claims 1, 13, 25 and 37, respectively, neither Zimmerman nor Lindhorst, either alone or in combination, teaches or suggests each and every feature of dependent claims 4-6, 10, 11, 16-18, 22, 23, 28-30, 34, 35 and 39. Accordingly, Applicants respectfully request

withdrawal of the rejection of dependent claims 4-6, 10, 11, 16-18, 22, 23, 28-30, 34, 35 and 39 under 35 U.S.C. § 103(a).

In addition to the above, neither Zimmerman nor Lindhorst, either alone in combination, teach or suggest the specific features of claims 10, 11, 22, 23, 34 and 35. Specifically, neither Zimmerman nor Lindhorst, either alone or in combination, teach or suggest an ability to edit a property using a custom editor interface. This feature is not taught by Zimmerman for the same reasons as noted above. Further, Lindhorst does not provide for the deficiencies of Zimmerman. That is, Lindhorst teaches nothing about custom editing as defined in the present invention. To the contrary, Lindhorst uses software components that are pre-existing modules, i.e. the String, Color and Number Dialog Boxes, that do not need to be modified to work within embodiments of the Lindhorst invention. There is nothing in Lindhorst that has anything to do with creating custom components or custom editors. Lindhorst merely uses methods defined by pre-existing components. Thus, neither Zimmerman nor Lindhorst, either alone or in combination, teach or suggest an ability to edit a property using a custom editor interface.

Since neither reference teaches or suggests the use of a custom editor interface, neither reference teaches or suggest that when a property editor is determined to use a customer editor interface, a graphical user interface includes a popup custom component area virtual button and at least one of a text entry field and an entry error indicator (claims 10, 22, 34). Similarly, neither reference teaches or suggests that such an entry error indicator is only displayed when an invalid entry in the text field entry area of the graphical user interface generated based on the fact that the property editor uses a custom editor interface (claims 11, 23, 35). Thus, in addition to being dependent upon independent claims 1, 13 and 25, claims 10, 11, 22, 23 34 and 35 are also allowable over the alleged combination of references by virtue of the specific features recited in these claims.

III. Conclusion

It is respectfully urged that the subject application is patentable over Zimmerman and Lindhorst and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: April 29, 2004

Stephen J. Walder, Jr.
Reg. No. 41,534
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 367-2001
Attorney for Applicants